

Corso formazione ReteIDRA

Savona 23/2 – 30/4 2010
corso avanzato

Alessandro Dentella



obiettivo

- Familiarizzare con il nostro sistema
- Capire come è strutturato
- Capire come parlargli ed ascoltarlo
- Se nel corso base possiamo dire che proviamo ad **usarlo** *prima* di averlo compreso, ora proviamo invece ad esplorare le cose che ci offre prima di averne bisogno. In modo poi da saperci muovere con scioltezza nel momento del bisogno.
- Essere a proprio agio quando ci capita di leggere documentazione e suggerimenti nei forum. Spesso questi son dati in forma testuale che è più facile da scrivere e più congeniale agli esperti.



Cosa conoscere

- Struttura cartelle/file
- File di configurazione: dove sono, come modificarli (editor)
- Conoscere l'installato (dpkg & Co.)
- Come collegarsi: ssh, con password, con chiave



Strumenti usati

- Questa lezione prevede che ogni persona abbia un pc su cui provare i comandi
- Il pc è in linea di massima una versione LTSP fat client su cui sarà fatta partire una macchina virtuale netkit prelevata dal server
- Da verificare se l'uso parallelo delle macchine virtuali e di LTSP intasa la rete. La maggior parte degli esercizi possono essere fatti in una qualunque macchina LTSP (senza netkit)



Il laboratorio

- La macchina virtuale parte da un laboratorio molto facile a cui faremo delle piccole modifiche:
 - Cambieremo il nome macchina
 - Cambieremo il modello
 - Elimineremo lo start di isi-setup



Comandi ed opzioni

- Per girovagare nel sistema daremo dei comandi dobbiamo quindi sapere quale è a struttura di un comando:

comando -opzione argomento1 arg2 arg3

- Le opzioni modificano il modo in cui il comando viene eseguito (es.: `grep`, `grep -i`), mentre gli argomenti sono in genere l'oggetto del comando (eg.: `grep root /etc/passwd`)



Uso interattivo della bash

- Usata interattivamente la bash ci permette di risparmiare battute usando il completamento:
 - Della prima parola
 - Degli argomenti



cp

- In generale, prima di fare modifiche ai file di configurazione, se non si è certi di quello che si fa è opportuno fare una copia. Il comando unix per fare una copia è `cp` che ha una comoda opzione ricorsiva: `-a` (studieremo poi un modo migliore la prossima lezione)
- Ogni comando è descritto nella sua pagina di manuale che si legge con il comando `man`: cosa fa `cp -v`? Cosa fa `cp -i`?
- Esercizio: sperimentare le opzioni di `cp`: `-a`, `-v`, `-f -i`



Ssh

- E' possibile gestire un sistema remoto loggandosi sul sistema remoto via `ssh`
- Ssh trasferisce i dati in modo crittato, i dati anche se letti da una persona che non sia il destinatario, non risulteranno comprensibili
- L' autenticazione può avvenire o con password o con scambio di chiavi pubblica/privata.
Esistenon client ssh sia per windows che per linux. Per windows si chiama putty, per linux si chiama ssh ed è installato di default in tutte le distribuzioni.



generazione chiavi

- Per generare le chiavi sotto linux occorre lanciare il comando:

`ssh-keygen -t dsa`

- Il comando genera una coppia di file pubblico/privato. Quello privato non deve essere dato a nessuno, quello pubblico può essere anche messo su un server web
- E` possibile entrare in un server senza usare la password, semplicemente dicendo ad ssh di usare la chiave se il server remoto ha la chiave pubblica accessibile
- Esercizio: connettersi al server, generare le chiavi e verificare che si riesce a connettersi senza password.



Studio server

- Ora che siamo connessi al server possiamo cominciare l'esplorazione del filesystem
- E` possibile farlo con nautilus per via grafica o con `cd` ed `ls` per via testuale
 - `cd`: change directory
 - `ls`: list file. La sua opzione più usata è `ls -l`, che mostra i dettagli del file, ora, proprietario e permessi (vedi allegato)



Studio pacchetti installati

- Potete vedere una presentazione [qui](#)
- repository in `/etc/apt/sources.list`
- `apt-get update`
- `apt-get upgrade`
- `apt-get install nome_pkg`
- `aptitude`
- `apt-get remove --purge`
- `dpkg -S nome_file`
- `dpkg -I pkg`
- `dpkg -L pkg`



sources

- I pacchetti vengono prelevati da alcune sorgenti normalmente via http. Le sorgenti sono suddivise per architettura (i386, amd64) e per codename (lenny, etch...) e component (main, free...)
- Ogni sorgente mette a disposizione un file Packages (firmato) con l'elenco dei pacchetti forniti dal repository.
- Un pacchetto potrebbe essere disponibile per una architettura e non per un'altra



dipendenze

- I pacchetti sono strutturati in un albero di dipendenze. Se isi dipende da python-isi, non sarà possibile installare isi se non è stato (già) installato python-isi.
- Apt-get è il comando che installa, ovvero informa il sistema di prelevare dalla sorgente che ha la versione più recente e di darlo a dpkg per la installazione in locale.
- Se apt-get vede che il pacchetto richiesto dipende da un altro pacchetto, procede all'installazione di tutte le dipendenze



Dipendenze non soddisfatte

- A volte le dipendenze non possono essere soddisfatte, normalmente per errore di chi ha creato il pacchetto o perchè le sorgenti non sono complete
- Forzare le dipendenze è una operazione che va fatta solo se si è sicuri che la cosa non comporti problemi
- Il comando 'dpkg -l' mostra tutti i pacchetti installati e la loro versione
- Esercizio: vediamo tutti i pacchetti installati sul sistema
- Vediamo tutti i pacchetti con nome isi



Trovare file

- Ci sono vari modi di cercare un file, ognuno ha il suo comando:
 - Dove sta il comando “ls”: `which ls`
il sistema cerca nel PATH per capire dove risiede il file usato
 - Quale pacchetto ha installato il comando “ls”?
`dpkg -S $(which ls)`
 - Qui `$()` significa: sostituisci qui il risultato di “which ls”. E` abbastanza probabile trovarlo nella documentazione



Find & locate

- find: trova un file con certe caratteristiche cercando nel filesystem
- locate: trova un file con un certo nome cercando in un database



grep

- Per potere analizzare velocemente l' output di un comando è spesso comodo usare il comando `grep` che deve essere concatenato all'altro per filtrarlo come ad esempio in:
`dpkg -l |grep isi`



FHS (seconda parte)

- I programmi devono essere installati in `/usr/*` o in `/opt` a seconda della struttura interna. I gestori li metteranno in `/usr`, gli installer dei programmi possono mettere in `/opt`
- `/var` è una cartella per dati rapidamente variabili (log, database, mail, ldap, proxy web, cache in genere...)
- `/tmp` è per file temporanei
- `/root` per la home del superuser
- `/lib` per librerie
- `/media` per



/etc

- Di gran lunga la cartella più interessante ed importante del sistema, /etc ha tutte le configurazioni ad eccezione di:
 - Configurazioni individuali (nella home)
 - Database ldap e database in genere se usati
- Molti file di configurazione sono leggibili da tutti, solo le password sono normalmente protette
- Esercizio: analizzare come è strutturata la cartella di configurazione di isi: dove sta?



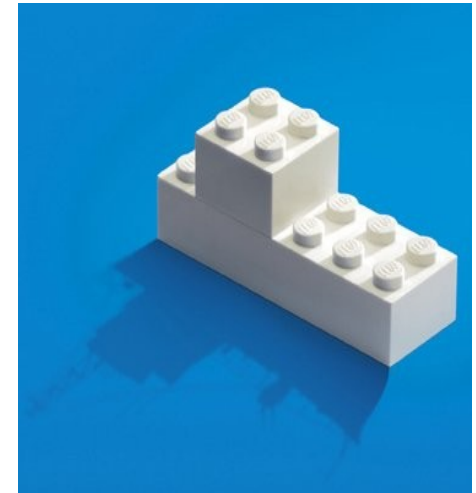
/home

- /home è la cartella dove vengono conservati tutti i dati individuali
- Non ci sono solo le home degli utenti, c'è ogni cartella da condividere. Normalmente è strutturata come partizione separata: come mai?



Comandi utili

- rm
- cat
- cut
- diff
- df
- du
- echo
- gzip/gunzip
- which/whereis
- head
- kill
- less
- passwd
- ps
- sort
- tail
- tar
- who
- wc



Esercizi

- Usare **tar** per creare una copia di backup di /etc e metterla in /tmp/\$user-etc.tgz dove \$user è il vostro nome.
- Scompattare lo stesso file e cercare se contiene un file chiamato
 - Slapcat
 - logon.conf
- Se li trovate contate quante righe hanno



- Contate quanti file ci sono in /etc a partire dalla copia di backup
- Contate quanti ce ne sono usando find
- Modificare un file con un editor (jmacs o nano – nell'allegato) e verificare con il comando diff come il sistema vede le modifiche:

```
cp /etc/fstab /tmp  
jmacs /tmp/fstab  
diff /etc/fstab /tmp
```



La shell bash

il collante che dai mattoncini ci permette di creare un programmino



intestazione

- Per scrivere una script di shell occorre
- `#!/bin/bash`
intestazione: informa il sistema che l'interprete per questi comandi è la shell `/bin/bash`
- Comandi come li daremmo da riga di comando, abbiamo anche a disposizione `if`, `for`, `while` e `case`:

```
if [ -f /etc/resolve.conf ] ; then  
    echo "Si c'è"  
fi
```



If nella bash

- Notare: if ; then fi (chiude if)
- test -f, per capire quali altri test possiamo fare:
help test
- Le parentesi quadre richiedono spazi prima e dopo!!!!
- Se volete potete aggiungere:
else
....
fi



For nella bash

- Un ciclo potrebbe essere così:

```
for file in /etc/passwd /etc/shadow; do  
    cp -b $file /home/backup  
done
```
- Notare: for ; do ; done
- File è il nome della variabile che contiene i nomi dei file /etc/passwd e poi /etc/shadow
- Per chiamare una variabile usiamo il \$: \$file
- Per assegnare una variabile:

```
file=/etc/passwd
```



Esercizi

- Guardare i file in `/etc/init.d/*` e vedere se capiamo la sintassi: usano costrutto:

```
case $a in  
    start) do_start  
        ;;  
    stop) do_stop  
        ;;
```

esac

- Usano anche `.file` che forza la lettura del file come codice. E` un modo di inglobare altre variabili/definizione di funzioni



Esercizi

- Creare una script che dica se esiste il file `/usr/share/netkit/fs/isi-lenny.img`, avvisare a schermo e con una mail (comando `mutt` o `mail`, quello che è presente)
- Creare una script che ritorni il pacchetto `debian` che contiene un comando dato come argomento
- Creare una script che “raddoppi” un file accodandolo due volte nell'output e che lo scriva se gli argomenti sono due



Argomenti in una script

- Gli argomenti di una script di shell vengono visti dall' interno come: \$1, \$2, \$3...

```
FILE=$1
```

```
if [ -f $FILE ] ; then echo ok; fi
```



Virtualizzazione con netkit



Savona -2010

Virtualizzazione a scopo educativo

- vmware è un software di virtualizzazione usato anche in sistemi di produzione. Per quanto affidabile non possiamo certo definirlo “leggero” ogni macchina virtuale chiede almeno 128 MB di RAM
- Per situazioni di studio è molto più pratico usare altre virtualizzazioni che facilitano il setup di tutta una rete, fatta di pc e switch

UML

- kernel linux che può essere eseguito come processo utente
- un processo user-mode-linux è chiamato virtual machine (vm), mentre la macchina su cui gira è chiamata macchina host
- molte macchine possono essere eseguite in contemporanea
- può eseguire solo linux su linux

vm

- ogni macchina virtuale “parte” da un filesystem contenuto in un file singolo
- può essere reso scrivibile tramite l'utilizzo della tecnica Copy-on-write (cow) che scrive solo le porzioni di file che cambiano. Il sistema che ne deriva usa poco spazio ma permette di creare tante linux box indipendenti

Vm (segue)

- ogni macchina virtuale richiede poche risorse: è quindi possibile eseguire svariate macchine in contemporanea
- ogni macchina ha una sua interfaccia di rete (o più)
- Le interfacce di rete possono essere connesse fra di loro tramite degli switch virtuali
- E` possibile decidere se devono vedersi fra di loro o meno



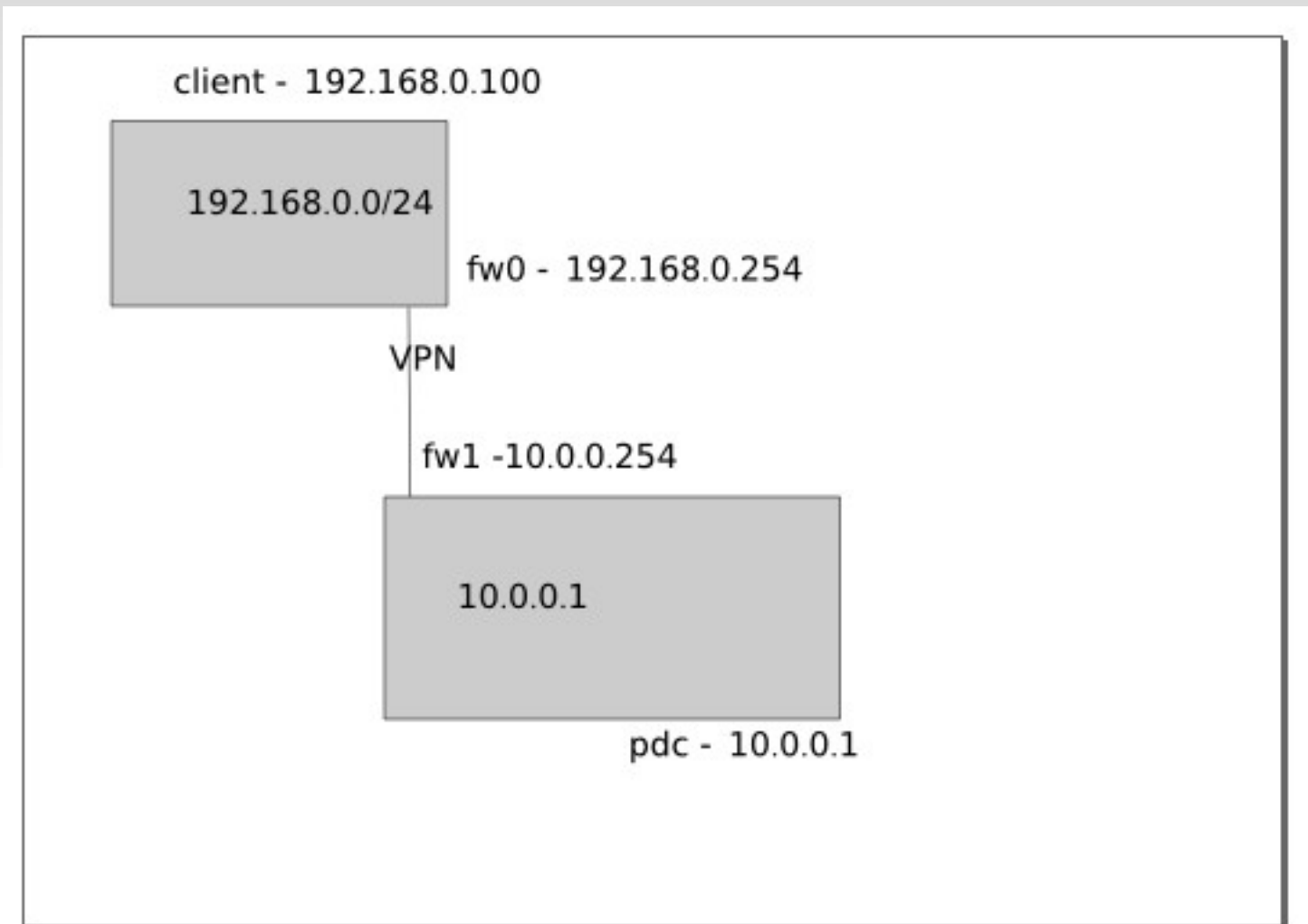
netkit

- Netkit è un sistema basato su UML che facilita il setup di un singolo pc ed in modo più eclatante di una rete
- fornisce alcuni comandi (script shell) che creano singole macchine, permettono di configurarle o fanno partire intere sottoreti, permettendo di salvare i risultati da una volta all'altra.
- E` sviluppato all'università roma3

Dominio di collisione

- cercare la definizione esatta
- ogni interfaccia può partecipare ad un dominio di collisione a scelta:
 - quello della macchina host
 - uno (o più) differenti
- è quindi possibile fare simulazioni di sottoreti differenti

Esempio



descrizione

- sede e plesso
- due reti
- due firewall
- 1 pc
- 1 pdc
- 3 domini di collisione:
 - sede
 - plesso
 - vpn



ping & arp

- dopo avere assegnato gli indirizzi proviamo a pingare le macchine.
- pingando una macchina inesistente vediamo passare i pacchetti ARP di richiesta del mac address

Cosa serve

- una immagine di rete is, modificata per potere essere utilizzata come nodo
- una pacchettizzazione di netkit facile da installare
- alcuni esercizi, ovvero configurazioni che fanno partire tutti i nodi che ci interessano per una particolare esercitazione.